Раздел IV. Информационные системы и технологии

УДК 004.94 DOI 10.18522/2311-3103-2020-2-193-200

В.Н. Гридин, В.И. Анисимов, С.А. Васильев

МЕТОДЫ ПОВЫШЕНИЯ ПРОИЗВОДИТЕЛЬНОСТИ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ*

Рассматриваются существующие и развивающиеся подходы построения современных веб-приложений. Определяются основные формы и направления развития современных веб-приложений, а также методы для повышения производительности обмена данными клиент-серверных системах. Освещаются разновидности и принципы установления каналов связи в распределенной клиент-серверной среде. Приводятся основные преимущества комбинированных методов взаимодействия с применением асинхронных полнодуплексных протоколов обмена данными для обеспечения высокой скорости передачи данных, предоставления информации своевременно, снижения нагрузки на серверную составляющую, снижения избыточности передаваемых данных. Указываются технологии децентрализации state management в одностраничных приложениях, взаимосвязь современных методик для обеспечения высокой степени интерактивности клиентской составляющей. Производится сравнительный анализ интеллектуального механизма обработки запросов, декларирования структуры данных и методов доступа к ним с ориентированным на работу с данными REST API, обеспечивающий различные вариации базовых CRUD операций. Освещаются основные достоинства подхода GraphQL по организации распределенного state management на основе предоставления клиентскому приложению графо-подобных структур неопределенного уровня вложенности, и возможности организации подписки на изменения в интересующем наборе данных. Приводятся проблемы традиционных систем хранения данных в современных информационных условиях, геометрическом накоплении сложно структурированных данных. Описываются основные подходы к хранению данных в разрезе концепции NoSQL. Рассматриваются преимущества использования модели ключ-значения в информационных системах. Определяются преимущества использования и принципы работы баз данных, использующих в качестве хранилища оперативную память. Рассматриваются недостатки указанных технологий хранения данных и предлагаются возможные пути их минимизации на основе коллаборации методов. В качестве выводов приводится схема зависимостей технологий эффективного обмена данными в современных веб-приложениях для обеспечения высокой степени интерактивности клиент-серверных веб-приложений.

Клиент-серверное приложение; WebSocket; GraphQL; SigmalR; NoSQL; BASE; ACID; Redis.

V.N. Gridin, V.I. Anisimov, S.A. Vasilev

METHODS OF IMPROVING PERFORMANCE OF MODERN WEB-APPLICATIONS

Existing and developing approaches to build modern web applications are considered. The main forms and directions of development modern web applications are determined, as well as methods for increasing the productivity of data exchange client-server systems. The varieties and principles of establishing communication channels in a distributed client-server environment are highlighted. The main advantages to use combined methods of interaction made with asynchronous full-duplex data exchange protocols are provided to ensure a high data transfer rate, provide

* Работа выполнена в рамках темы № 0071-2019-0001.

information in a timely manner, reduce the load on the server component, and reduce the redundancy of transmitted data. The technologies of state management decentralization in single-page applications, the interconnection of modern technologies to ensure a high degree of interactivity of the client component are indicated. A comparative analysis of the intelligent query processing mechanism, declaring the data structure and access methods with data-oriented REST APIs is carried out, which provides variations of the basic CRUD operations. The main advantages of GraphQL's approach to organize distributed state management based on providing graph-like structures with an indefinite level of nesting to the client application and the possibility of subscribing to change in the data set of interest are highlighted. The problems of traditional data storage systems in modern information conditions, the geometric accumulation of complex structured data are presented. The basic approaches to data storage in the context of the NoSQL concept are described. The advantages of using the key-value model in information systems and the principles of operation databases using RAM as storage are considered. The disadvantages of these technologies for data storage are considered and possible ways to minimize them based on a collaboration of methods are proposed. Thus, a dependency diagram of technologies for efficient data exchange in modern web applications is provided to ensure a high degree of interactivity client-server web applications.

Client-server application; WebSocket; GraphQL; SigmalR; NoSQL; BASE; ACID; Redis.

Введение. В настоящее время наиболее распространён вид предоставления сервисов и услуг посредством веб-приложений через сеть интернет. Существуют два принципа построения веб-приложений – традиционный и одностраничный (SPA – Single Page Application и PWA – Progressive Web Application). В первом случае бизнес логика выполняется на серверной составляющей, основные преимущества такого подхода весьма архаичны, это минимальные технические требования к клиентской стороне, к hardware и software вплоть до отсутствия поддержки JavaScript. Одностраничные приложения являются более требовательными к клиентской составляющей, как с точки зрения hardware, так и software несмотря на то, что основная бизнес логика может быть также размещена на серверной части, интерфейс целиком определяется и формируется на клиентской стороне, используя сервер как поставщик данных, заполняющих сформировавшийся шаблон. Сохранение состояния сервиса, предоставляемого пользователю, может осуществляться именно на клиентской стороне, при этом взаимодействие с сервером осуществляется главным образом через веб-API. Также имеются гибридные вариации приложений, например Blazor, однако на текущий момент они не являются востребованными [1].

Современное восприятие пользовательского интерфейса сдвигает парадигму построения веб-приложений в сторону одностраничной модели. Коммерческая конкурентоспособность такого метода построения веб-приложений достигается путем имитации поведения нативного программного обеспечения. В некоторых случаях веб-приложение преобразуется в нативное программное обеспечение, ярким примером разработки графических приложений для основных операционных систем (MacOS, Windows, Linux) на основе веб-технологий является фреймворк Electron, способный к рендеренгу html, css и JavaScript на вышеуказанных платформах. Для обеспечения такой работы клиентской стороне необходим высокоскоростной обмен данными, который может быть достигнут путем эффективной подготовки данных на сервере, а также быстрым каналом связи, обеспечивающим своевременной информацией как клиентскую, так и серверную составляющую [2].

Каналы связи. В качестве канала связи рекомендуется использовать полнодуплексный протокол, обеспечивающий своевременную доставку данных клиенту. В случае выполнения ресурсоемких вычислений, сервер инициирует отправку данных по их готовности. Такой подход повышает эффективность канала связи, а также снижает нагрузку на сервер, т.к. клиентское приложение ожидает ответ в отличии от стандартной модели, циклично формирующей запросы на сервер с заданным интервалом до момента получения результата. В качестве примера можно

привести протокол WebSocket, отвечающий вышеуказанным требованиям, в дополнении к этому следует отметить, что производительность данного канала связи выше в сравнении с протоколом HTTP, используемым в традиционной модели за счет существенного снижения объема служебной информации в сообщениях, например, отсутствия заголовков сообщений. Для организации работы WebSocket соединения требуется лишь один HTTP-запрос и HTTP-ответ (рукопожатие) для установления связи. Впоследствии передаваемые данные ограничиваются всего двумя байтами служебной информации при каждом запросе и ответе соответственно.

Преимуществом использования протокола WebSocket является увеличение продолжительности жизни канала связи, который может существовать значительно дольше в сравнении с другими псевдо-асинхронными технологиями из-за особенностей формата запросов и алгоритмов фильтрации прокси-серверов. Соответственно соединение может находиться в открытом состоянии сколь угодно долго и быть неактивным, а значит не требовать ресурсов для обработки. При этом в любой момент времени имеется возможность передавать данные по заранее созданному соединению.

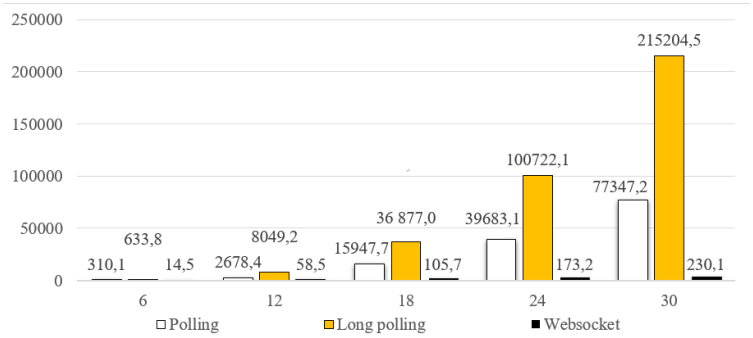

Рис. 1. Зависимость объема переданных данных при использовании различных подходов от времени

Для оценки эффективности полнодуплексной асинхронной технологии WebSocket по отношению к другим псевдо-асинхронным технологиям проведено испытание по сравнению количества передаваемых пакетов для поддержания актуальной информации в одностраничном приложении. В течении 30 минут производилось наблюдение следующих процессов: формирование данных на сервере каждые 5 минут, отправление polling запроса каждые 3 минуты, измерение трафика, проходящего через сервер каждые 6 минут. На рис. 1 представлена диаграмма, отражающая зависимость количества переданных пакетов от времени.

Стоит отметить, что использование протокола WebSocket налагает обязательства на построение клиент-серверных систем, например, обязательная поддержка технологии WebSocket браузерами, в качестве альтернативного подхода следует рассмотреть решение SignalR, обеспечивающее наиболее эффективный транспорт из доступных в конкретной ситуации, в этот пакет входят WebSocket, Events и LongPolling [3].

Подготовка данных. К эффективной подготовке данных относится их хранение на стороне сервера, т.е. насколько быстро их можно получить по запросу. Сперва следует уделить внимание технологиям middleware, оптимизирующим доступ и компоновку данных. Стандартным решением для одностраничных веб-приложений является REST API, отличительная черта которого – ориентация на работу с данными с обеспечением различных вариаций базовых CRUD (create,

read, update, delete) операций. Каждый endpoint (точка входа) позволяет провести одну или несколько указанных операций над определенным набором данных. Узкое место при использовании данной концепции проявляется в случаях необходимости получения части определенного набора данных или измененного набора данных. Решения – два, в первом случае клиентское приложение получает избыточный набор данных, нагружая базу данных и канал связи, увеличивая скорость реакции клиентской части для пользователя; во-втором случае производится дублирование endpoint на сервере с изменениями, обеспечивающими передачу необходимого набора данных, но усложняющих модернизацию, эксплуатацию и сопровождение серверной составляющей [4].

Для обеспечения эффективной работы endpoint, отправки данных на клиентское приложение без избытков и отсутствия дублирования самих endpoint, следует использовать интеллектуальный механизм декларирования структуры данных и методов доступа к ним. Для достижения описанного результата клиентское приложение должно отправить специальный запрос на именной endpoint описывающий необходимые данные. Сервер при получении такого запроса применяет resolver функцию, обеспечивающую поиск, изменение и компоновку необходимых данных для отправки их на клиентскую составляющую. Механизмом работы с запросами по данному принципу является GraphQL, выпущенный компанией Facebook в 2015 году для обеспечения эффективной обработки запросов своей широкой аудитории. На текущий момент существует большое количество библиотек для различных платформ, обеспечивающих интеллектуальную работу с запросами на сервере. Одна из отличительных особенностей подхода GraphQL - наличие возможности обеспечения ответа на запросы с подзапросами за счет их рекурсивной обработки с учетом возможности добавления аргументов в самих запросах, предоставляя клиентскому приложению графо-подобных структур неопределенного уровня вложенности [5]. Также преимуществом является возможность мутации данных для обеспечения всех CRUD операций на одном endpoint. При этом существует механизм организации подписки на данные – постоянной связи между клиентским и серверным приложениями, реагирующий на изменения в указанном наборе данных. В совокупности своих функций подход GraphQL позволяет отказаться от использования state management на клиентской стороне для SPA или существенно снизить масштаб реализации для PWA приложений. Существуют специальные библиотеки, способные работать с децентрализованным state management на клиентской и серверной составляющих, например Loona, представляющие связующее звено для sate management на сервере в виде подписок на структуру данных GraphQL и локального state management Apollo на клиентской стороне, для распространенных фреймворков Angular и React специализирующихся на создании SPA и PWA приложений. Также следует отметить, что использование данного middleware позволяет поддерживать взаимодействие с неограниченным количеством источников данных [6].

Хранение данных. К эффективной подготовке данных относятся непосредственно средства хранения – базы данных. Начало 2000-х годов было ознаменовано теоремой CAP (Принцип был предложен Эриком Брюером), согласно которой распределённая система может обеспечить не более двух принципов из трех: согласованность (Consistency), доступность (Availability) и устойчивость к разделению (Partition tolerance), и чуть позже появлению термина Big Data, определяющего данные с тремя признаками: отсутствие структурированности, гипертрофированность физического объема и наличие динамики их прироста. Эти события привели к необходимости изменения систем хранения данных в пользу технологий, обеспечивающих высокую степень масштабируемости (Partition tolerance), являю-

щейся слабой стороной реляционной концепции наряду с модификацией её схемы ввиду строгой структурированности данных. Впоследствии появляется новый подход к построению распределенных систем BASE (Basically Available, Soft-state, Eventually consistent) – базовая доступность, неустойчивое состояние, согласованность в конечном счёте. Этот подход описывает свойства присущие NoSQL базам данных, основной целью которых является решение проблемы масштабирования данных и доступность в той или иной степени за счет атомарности и согласованности [7].

Подходы NoSQL также имеют несколько разновидностей: хранилище ключ-значение, документно-ориентированное, колоночное хранилище, хранилище на основе графов. Большего внимания в отношении информационных систем заслуживает разделяющая современный формат обмена данными в сети интернет, модель ключ-значение. При запросе клиентское приложение может получать объект данных (слабоструктурированные данные), целиком отвечающий решению конкретной задачи, при этом достигается высокая скорость предоставления информации по ключу, уменьшается количество запросов для получения разных структур данных, т.к. они не разделены по таблицам, а находятся внутри объекта, уменьшается объем передаваемых данных при условии отсутствия какого-либо свойства объекта эта информация исключается из самого объекта (ключ и значение) [8].

Следующая стадия оптимизации подготовки данных – использование баз данных, работающих с оперативной памятью (In-memory). Не теряя всех вышеописанных преимуществ NoSQL подхода, использование In-memory баз данных позволяет добиться увеличения скорости выполнения запроса до пяти порядков, при этом обеспечивает сохранность данных в условиях нестабильной работы сервера, отключения питания и т.п. Несмотря на то, что данные хранятся в оперативной памяти и доступ к ним осуществляется предельно быстро, транзакции (операции изменяющие данные) дополнительно дублируется в лог в ПЗУ, обеспечивая их сохранность. Для оптимизации лога транзакций производятся периодические снимки состояния (snapshot), являющиеся копией базы данных, позволяющие очищать лог транзакций до момента времени предшествующему снимку. Ярким примером данного решения может служить резидентная система управления базами данных Redis, являющаяся наиболее востребованной на текущий момент по версии ресурса db-engines [9].

Подход NoSQL также имеет и ряд недостатков, в первую очередь – это отсутствие возможности соответствия требования ACID (атомарность, согласованность, изолированность) к транзакциям. Существуют попытки создания оптимальной системы хранения данных, сочетающей преимущества обоих подходов (NewSQL), однако на текущий момент они не увенчались успехом. В случае решения задачи обеспечения высокой производительности для ряда данных и надежности транзакций для другой группы данных следует применять коллаборативные решения на основе традиционного и описанного подходов.

Вывод. Для поддержания коммерческой конкурентоспособности современных веб-приложений следует обеспечить высокую степень интерактивности их интерфейса на уровне нативного приложения. Чтобы обеспечить такой уровень быстродействия необходимо иметь эффективно организованный распределенный state management для снижения нагрузки на клиентское приложение. Для обеспечения своевременной поставки необходимой информации, по мере появления данных, без избыточности следует использовать полнодуплексные или комбинированные подходы для установления каналов связи. Сопряжение указанных подходов с интеллектуальным механизмом декларирования структуры данных и методов доступа к ним, предоставляющих возможность установления подписки на необходимые наборы данных, а также формирование графо-подобных структур не-

определенного уровня вложенности, позволяют добиться высоко уровня производительности обмена данными. Для обеспечения эффективности работы с информацией приведенный middleware должен предусматривать работу с колаборативными масштабируемыми системами хранения слабосвязанных данных, использующих в качестве структурных единиц объектную модель, а в качестве носителя информации оперативное запоминающее устройство.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. *Валитова Н.Л., Кремлева Э.Ш., Кашафутдинов Р.К.* Перспективы применения технологии pwa для расширения e-learning систем на мобильные платформы // Образовательные технологии и общество. – 2020. – № 1. – С. 115-124.
2. *Гавриленко Ю.Ю., Саада Д.Ф., Ильюшин Е.А., Намиот Д.Е.* Разработка прогрессивного web-приложения для системы управления push-уведомлениями // International Journal of Open Information Technologies. – 2018. – № 9.
3. *Долгов А.Н., Нуруллин Р.Ю.* Программная платформа node. JS // Достижения науки и образования. – 2016. – № 12 (13).
4. *Пустобаев А.И.* О сервисе рассылки push-уведомлений // International Journal of Open Information Technologies. – 2015. – № 6. – С. 13-20.
5. *Ильин Д.Ю., Никульчев Е.В., Колясников П.В.* Выбор технологических решений для разработки программного обеспечения распределенных информационных систем // Современные информационные технологии и ИТ-образование. – 2018. – № 2. – С. 344-354
6. *Анисимов В.И., Васильев С.А., Гридин В.Н.* Высокоскоростной полнодуплексный метод обмена данными для распределенных САПР // Известия ЮФУ. Технические науки. – 2018. – № 4 (198). – С. 38-47.
7. *Фереферов Е.С., Бычков И.В., Хмельнов А.Е.* Технология разработки приложений баз данных на основе декларативных спецификаций // Вычислительные технологии. – 2014. – № 5. – С. 85-100.
8. *Анисимов В.И., Васильев С.А., Тарасова О.Б., Чернов А.Н.* Сравнительный обзор методологий для создания интерактивных веб-приложений // Тр. конгресса по интеллектуальным системам и информационным технологиям «IS&IT'18». – 2018. – Т. 1. – С. 36-39.
9. *Игнатьев А.Ю.* Обзор технологии GraphQL // Молодой ученый. – 2019. – № 15 (253). – С. 22-24.
10. *Бодров М.Ю.* Современный подход к разработке одностраничных веб приложений // Современные тенденции развития науки и технологий. – 2015. – № 3-1. – С. 43-45.
11. *Erik Wittern, Alan Cha1, James C. Davis, Guillaume Baudart, Louis Mandel*. An Empirical Study of GraphQL Schemas // ICSOC19. – 2019. – P. 1-16.
12. *Романенко Е.В.* Место Big Data в современной социально-экономической жизни общества // Инновационная наука. – 2016. – № 4-3 (16). – С. 143-145.
13. *Абдыкаримова А.Т.* Big Data: проблемы и технологии // Международный журнал гуманитарных и естественных наук. – 2019. – № 5-1. – С. 16-18.
14. *Brewer Eric A.* A Certain Freedom: Thoughts on the CAP Theorem (англ.) // Proceeding of the XXIX ACM SIGACT-SIGOPS symposium on Principles of distributed computing. – N. Y.: ACM, 2010.
15. *Новиков Б.А*. Сравнительный анализ производительности SQL И NOSQL СУБД // КИО. – 2017. – № 4. – С. 48-63.
16. *Прамодкумар Дж. Садаладж, Мартин Фаулер*. NoSQL. Новая методология разработки нереляционных баз данных. – Вильямс, 2017.
17. *Савоськин И.В., Фирсов А.О*. Исследование способов применения nosql и реляционных баз данных // E-Scio. – 2019. – № 6 (33). – С. 41-49.
18. *Якушин А.Ю., Муковозов А.М., Исмоилов М.И.* Сравнительный анализ реляционной базы данных и документоориентированной NoSQL базы данных в разрезе их применения при создании локального чата/мессенджера // Инновационная наука. – 2018. – № 4. – С. 73-82.
19. DB-Engines Ranking of Key-value Stores // https://db-engines.com URL: https://db-engines.com/en/ranking/key-value+store (дата обращения: 12.05.2020).
20. *Соболь А.С.* Построение и адаптация NewSQL СУБД в частном «Облаке» // Сибирский журнал науки и технологий. – 2013. – № 4 (50). – С. 75-80.

REFERENCES

1. *Valitova N.L., Kremleva E.Sh., Kashafutdinov R.K.* Perspektivy primeneniya tekhnologii pwa dlya rasshireniya e-learning sistem na mobil'nye platformy [Prospects for applying pwa technology to expand e-learning systems to mobile platforms], *Obrazovatel'nye tekhnologii i obshchestvo* [Educational technologies and society], 2020, No. 1, pp. 115-124.
2. *Gavrilenko Yu.Yu., Saada D.F., Il'yushin E.A., Namiot D.E*. Razrabotka progressivnogo web-prilozheniya dlya sistemy upravleniya push-uvedomleniyami [Development of a progressive web application for push notification management system], *International Journal of Open Information Technologies*, 2018, No. 9.
3. *Dolgov A.N., Nurullin R.Yu.* Programmnaya platforma node. JS [The software platform of the node. JS], *Dostizheniya nauki i obrazovaniya* [Achievements of science and education], 2016, No. 12 (13).
4. *Pustobaev A.I.* O servise rassylki push-uvedomleniy [About the push notification mailing service], *International Journal of Open Information Technologies*, 2015, No. 6, pp. 13-20.
5. *Il'in D.Yu., Nikul'chev E.V., Kolyasnikov P.V.* Vybor tekhnologicheskikh resheniy dlya razrabotki programmnogo obespecheniya raspredelennykh informatsionnykh sistem [Selection of technological solutions for software development of distributed information systems], *Sovremennye informatsionnye tekhnologii i IT-obrazovanie* [Modern information technologies and IT education], 2018, No. 2, pp. 344-354.
6. *Anisimov V.I., Vasil'ev S.A., Gridin V.N.* Vysokoskorostnoy polnodupleksnyy metod obmena dannymi dlya raspredelennykh SAPR [High-speed full-duplex data exchange method for distributed CAD systems], *Izvestiya YuFU. Tekhnicheskie nauki* [Izvestiya SFedU. Engineering Sciences], 2018, No. 4 (198), pp. 38-47.
7. *Fereferov E.S., Bychkov I.V., Khmel'nov A.E.* Tekhnologiya razrabotki prilozheniy baz dannykh na osnove deklarativnykh spetsifikatsiy [Technology for developing database applications based on declarative specifications], *Vychislitel'nye tekhnologii* [Computing technologies], 2014, No. 5, pp. 85-100.
8. *Anisimov V.I., Vasil'ev S.A., Tarasova O.B., Chernov A.N.* Sravnitel'nyy obzor metodologiy dlya sozdaniya interaktivnykh veb-prilozheniy [Comparative review of methodologies for creating interactive web applications], *Tr. kongressa po intellektual'nym sistemam i informatsionnym tekhnologiyam «IS&IT'18»* [Proceedings of the Congress on intelligent systems and information technologies "IS&IT'18"], 2018, Vol. 1, pp. 36-39.
9. *Ignat'ev A.Yu.* Obzor tekhnologii GraphQL [Overview of GraphQL technology], *Molodoy uchenyy* [Young scientist], 2019, No. 15 (253), pp. 22-24.
10. *Bodrov M.Yu.* Sovremennyy podkhod k razrabotke odnostranichnykh veb prilozheniy [Modern approach to developing single page web applications], *Sovremennye tendentsii razvitiya nauki i tekhnologiy* [Modern trends in the development of science and technology], 2015, No. 3-1, pp. 43-45.
11. *Erik Wittern, Alan Cha1, James C. Davis, Guillaume Baudart, Louis Mandel*. An Empirical Study of GraphQL Schemas, *ICSOC19*, 2019, pp. 1-16.
12. *Romanenko E.V.* Mesto Big Data v sovremennoy sotsial'no-ekonomicheskoy zhizni obshchestva [The place of Big Data in the modern socio-economic life of society], *Innovatsionnaya nauka* [Innovative science], 2016, No. 4-3 (16), pp. 143-145.
13. *Abdykarimova A.T.* Big Data: problemy i tekhnologii [Big Data: problems and technologies], *Mezhdunarodnyy zhurnal gumanitarnykh i estestvennykh nauk* [International journal of Humanities and natural Sciences], 2019, No. 5-1, pp. 16-18.
14. *Brewer Eric A.* A Certain Freedom: Thoughts on the CAP Theorem (англ.), *Proceeding of the XXIX ACM SIGACT-SIGOPS symposium on Principles of distributed computing*. N. Y.: ACM, 2010.
15. *Novikov B.A.* Sravnitel'nyy analiz proizvoditel'nosti SQL I NOSQL SUBD [Comparative analysis of SQL and NOSQL DBMS performance], *KIO* [KIO], 2017, No. 4, pp. 48-63.
16. *Pramodkumar Dzh. Sadaladzh, Martin Fauler*. NoSQL. Novaya metodologiya razrabotki nerelyatsionnykh baz dannykh [NoSQL. A new methodology for developing non-relational databases]. Vil'yams, 2017.
17. *Savos'kin I.V., Firsov A.O.* issledovanie sposobov primeneniya nosql i relyatsionnykh baz dannykh [Research of ways to use nosql and relational databases], *E-Scio*, 2019, No. 6 (33), pp. 41-49.

18. *Yakushin A.Yu., Mukovozov A.M., Ismoilov M.I.* Sravnitel'nyy analiz relyatsionnoy bazy dannykh i dokumentoorientirovannoy NoSQL bazy dannykh v razreze ikh primeneniya pri sozdanii lokal'nogo chata/messendzhera [Comparative analysis of a relational database and a document-oriented NoSQL database in terms of their use in creating a local chat/messenger], *Innovatsionnaya nauka* [Innovative science], 2018, No. 4, pp. 73-82.
19. DB-Engines Ranking of Key-value Stores. Available at: https://db-engines.com URL: https://db-engines.com/en/ranking/key-value+store (accessed 12 May 2020).
20. Sobol' A.S. Postroenie i adaptatsiya NewSQL SUBD v chastnom «Oblake» [Building and adapting a NewSQL DBMS in a private "Cloud"], *Sibirskiy zhurnal nauki i tekhnologiy* [Siberian journal of science and technology], 2013, No. 4 (50), pp. 75-80.

Статью рекомендовал к опубликованию д.ф.-м.н. А.С. Бугаев.

Гридин Владимир Николаевич – ФГБУН Центр информационных технологий в проектировании РАН; e-mail: info@ditc.ras.ru; 143003, Одинцово, ул. маршала Бирюзова, 7а; тел.: +74955960219; д.т.н.; профессор; научный руководитель.

Анисимов Владимир Иванович – Санкт-Петербургский государственный электротехнический университет; e-mail: vianisimov@inbox.ru; 197376, Санкт-Петербург, ул. Профессора Попова, 5; тел.: +78122343675; д.т.н.; профессор; г.н.с.

Васильев Сергей Алексеевич – e-mail: venom-gt@list.ru; к.т.н.; ассистент.

Gridin Vladimir Nikolayevich – Design Information Technology Center RAS (DITC RAS); e-mail: info@ditc.ras.ru; 7a, Marshal Biryuzov street, Odintsovo, 143003, Russia; phone: +74955960219; dr. of eng. sc.; professor; scientific director.

Anisimov Vladimir Ivanovic – Saint Petersburg State Electrotechnical University; e-mail: vianisimov@inbox.ru; 5, Prof. Popov street, Saint Petersburg, 197376, Russia; phone: +78122343675; dr. of eng. sc.; professor; chief researcher.

Vasilev Sergey Alexievich – e-mail: venom-gt@list.ru; assistant.

УДК 004.051 DOI 10.18522/2311-3103-2020-2-200-209

Д.Е. Чикрин, А.А. Егорчев, Д.В. Ермаков

МЕТОДОЛОГИЯ S.M.A.R.T.E.S.T. H-GQM ДЛЯ КОНТРОЛИРУЕМОЙ ЭВОЛЮЦИИ СИСТЕМ ADAS

Вывод на массовый рынок транспортных средств (легковых и грузовых автомобилей) с высокой степенью автоматизации – уровня ADAS 3+ – ожидается с начала 2020-х годов. На текущий момент абсолютным большинством крупных автопроизводителей ведутся исследования и разработки в данном направлении, достаточно большое количество прототипов, предсерийных и серийных систем[1] уже продемонстрировано. Системы автоматизированного управления автомобилем – ADAS (advanced driver assistance systems) – представляют собой сложные аппаратно-программные комплексы, особенность которых состоит в неизменности ядра аппаратной платформы на протяжении одного или нескольких поколений автомобилей. При этом требуется обеспечить возможность обновления (эволюции) системы для исправления ошибок и расширения функциональности, особенно в условиях активно развивающихся сенсорных периферийных систем и программных алгоритмов. Для оценки и сопровождения разработки сложных систем применяется методология GQM (Goal, Question, Metric – цель, вопрос, метрика) и её модификации. Однако, область их применения ограничена исключительно программными продуктами; также не рассматриваются явно вопросы применения методологии GQM для анализа и сопровождения процессов эволюции сложных технических систем. В статье предлагается методоло-

[1] Tesla, Audi, Volvo, Komatsu и другие производители.